**[Smart Surveying Device For Resource Optimization](http://www.iotgyan.com/iot-projects/smart-surveying-device-for-resource-optimization)**

**Abstract:**

This device can be integrated in the place where we want to monitor different parameters. By using this data the authorities can take the necessary steps and make efficient arrangements which will be helpful for their employees.  Features:  Counting the number of people entering the conference hall and monitoring their time patterns.

* Counting the number of vehicles entering the parking area and monitoring their time patterns.
* Crowd estimation in an area.

**Problem Statement:**

In corporate offices and other industries there are some common resources which are used by different people in that industry. Sometimes there will be the chances that the resources would be more or insufficient. This would be the drawback.

**Project working process:**

Face Detection is a computer vision technology that helps to locate/visualize human faces in digital images. This technique is a specific use case of object detection technology that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.

**Components used:**

**Nodemcu specifications:**

* Wi-Fi Module – ESP-12E module similar to [ESP-12](https://www.aliexpress.com/item/new-esp8266-ESP-12-wifi-module-ESP8266-serial-WIFI-coexistence-module-AP-STA-AP-STA-WIFI/32239125397.html) module but with 6 extra GPIOs.
* USB – micro USB port for power, programming and debugging
* Headers – 2x 2.54mm 15-pin header with access to GPIOs, SPI, UART, ADC, and power pins
* Misc – Reset and Flash buttons
* Power – 5V via micro USB port
* Dimensions – 49 x 24.5 x 13mm

**Oled specifications:**

* Improved image quality - better contrast, higher brightness, fuller viewing angle, a wider color range and much faster refresh rates.
* Lower power consumption.
* Simpler design that enables ultra-thin, flexible, foldable and transparent displays
* Better durability - OLEDs are very durable and can operate in a broader temperature range

**Source code:**

# USAGE  
# python real\_time\_object\_detection.py --prototxt MobileNetSSD\_deploy.prototxt.txt --model MobileNetSSD\_deploy.caffemodel  
  
# import the necessary packages  
from imutils.video import VideoStream  
from imutils.video import FPS  
import numpy as np  
import argparse  
import imutils  
import time  
import cv2  
import requests  
count=0

# construct the argument parse and parse the arguments  
ap = argparse.ArgumentParser()  
ap.add\_argument("-p", "--prototxt", required=True,  
help="path to Caffe 'deploy' prototxt file")  
ap.add\_argument("-m", "--model", required=True,  
help="path to Caffe pre-trained model")  
ap.add\_argument("-c", "--confidence", type=float, default=0.2,  
help="minimum probability to filter weak detections")  
args = vars(ap.parse\_args())  
  
# initialize the list of class labels MobileNet SSD was trained to  
# detect, then generate a set of bounding box colors for each class   
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",  
"bottle", "bus", "car", "cat", "chair", "cow", "diningtable",  
"dog", "horse", "motorbike", "person", "pottedplant", "sheep",  
"sofa", "train", "tvmonitor"]  
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))  
  
# load our serialized model from disk  
print("[INFO] loading model...")  
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])  
  
# initialize the video stream, allow the cammera sensor to warmup,  
# and initialize the FPS counter  
print("[INFO] starting video stream...")  
vs = VideoStream(src=0).start()  
time.sleep(2.0)  
fps = FPS().start()  
  
# loop over the frames from the video stream  
while True:  
# grab the frame from the threaded video stream and resize it  
# to have a maximum width of 400 pixels  
frame = vs.read()  
frame = imutils.resize(frame, width=800)  
  
# grab the frame dimensions and convert it to a blob  
(h, w) = frame.shape[:2]  
blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),  
0.007843, (300, 300), 127.5)  
  
# pass the blob through the network and obtain the detections and  
# predictions  
net.setInput(blob)  
detections = net.forward()  
#for j in range(19):

# loop over the detections  
for i in np.arange(0, detections.shape[2]):  
# extract the confidence (i.e., probability) associated with  
# the prediction  
confidence = detections[0, 0, i, 2]  
                  
# filter out weak detections by ensuring the `confidence` is  
# greater than the minimum confidence  
if confidence > args["confidence"]:  
# extract the index of the class label from the  
# `detections`, then compute the (x, y)-coordinates of  
# the bounding box for the object  
idx = int(detections[0, 0, i, 1])  
box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])  
(startX, startY, endX, endY) = box.astype("int")  
                          
# draw the prediction on the frame  
label = "{}: {:.2f}%".format(CLASSES[idx],  
confidence \* 100)  
cv2.rectangle(frame, (startX, startY), (endX, endY),  
COLORS[idx], 2)  
y = startY - 15 if startY - 15 > 15 else startY + 15  
cv2.putText(frame, label, (startX, y),  
cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, COLORS[idx], 2)  
  
# show the output frame  
cv2.imshow("Frame", frame)  
  
key = cv2.waitKey(1) & 0xFF  
          
# if the `q` key was pressed, break from the loop  
if key == ord("q"):  
break  
#counting  
count=i  
print ('count is',count)  
#url  
url='[https://sip-iot-app1.eu-gb.mybluemix.net/command?message='+str(count)](https://sip-iot-app1.eu-gb.mybluemix.net/command?message=%27+str(count))  
r=requests.get(url)  
print(r.text)  
# update the FPS counter  
fps.update()  
  
# stop the timer and display FPS information  
fps.stop()  
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))  
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))  
  
# do a bit of cleanup  
cv2.destroyAllWindows()  
vs.stop()

**Coding to display the count on oled:**

#include <ESP8266WiFi.h>

const char\* ssid = "chi";

const char\* password = "navyanavya";

const char\* host = "https://sip-iot-app1.eu-gb.mybluemix.net";

#include <Wire.h>

#include <Adafruit\_SSD1306.h>

#include <Adafruit\_GFX.h>

#define SSD1306\_LCDHEIGHT 64

// OLED display TWI address

#define OLED\_ADDR 0x3C

Adafruit\_SSD1306 display(-1);

#if (SSD1306\_LCDHEIGHT != 64)

#error("Height incorrect, please fix Adafruit\_SSD1306.h!");

#endif

void setup() {

Serial.begin(115200);

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("");

Serial.println("WiFi connected");

Serial.println("IP address: ");

Serial.println(WiFi.localIP());

}

void loop() {

Serial.print("connecting to ");

Serial.println(host);

// Use WiFiClient class to create TCP connections

WiFiClient client;

const int httpPort = 80;

if (!client.connect(host, httpPort)) {

Serial.println("connection failed");

return;

}

// We now create a URI for the request

String url ="1";

Serial.print("Requesting URL: ");

Serial.println(url);

// This will send the request to the server

client.print(String("GET ") + url + " HTTP/1.1\r\n" +

"Host: " + host + "\r\n" +

"Connection: close\r\n\r\n");

delay(10);

// Read all the lines of the reply from server and print them to Serial

while(client.available()){

String line = client.readStringUntil('\r');

Serial.print("line");

Serial.print(line);

}

delay(1000);

Serial.println();

Serial.println("closing connection");

delay(4000);

display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR);

display.clearDisplay();

// display a line of text

display.setTextSize(1);

display.setTextColor(BLACK,WHITE);

display.setCursor(0,10);

display.print(url);

// update display with all of the above graphics

display.display();

}

**Software Installations:**

**Python installation:**

**Step 1:** Double-click the icon labeling the file **python**-3.7.0.exe. An Open File - Security Warning pop-up **window** will appear.

**Step 2:** Click Run. A **Python** 3.7.0 (32-bit) Setup pop-up **window** will appear. ...

**Step 3 :** Highlight the **Install** Now (or Upgrade Now) message, and then click it. ...

**Step 4 :** Click the Yes button. ...

**Step 5 :**Click the Close button.

**Opencv installation:**

1. Below Python packages are to be downloaded and installed to their default locations.
   1. [Python-2.7.x](http://python.org/ftp/python/2.7.5/python-2.7.5.msi).
   2. [Numpy](http://sourceforge.net/projects/numpy/files/NumPy/1.7.1/numpy-1.7.1-win32-superpack-python2.7.exe/download).
   3. [Matplotlib](https://downloads.sourceforge.net/project/matplotlib/matplotlib/matplotlib-1.3.0/matplotlib-1.3.0.win32-py2.7.exe) (Matplotlib is optional, but recommended since we use it a lot in our tutorials).
2. Install all packages into their default locations. Python will be installed to C:/Python27/.
3. After installation, open Python IDLE. Enter import numpy and make sure Numpy is working fine.
4. Download latest OpenCV release from [sourceforge site](http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.6/OpenCV-2.4.6.0.exe/download) and double-click to extract it.
5. Goto opencv/build/python/2.7 folder.
6. Copy cv2.pyd to C:/Python27/lib/site-packages.
7. Open Python IDLE and type following codes in Python terminal.

1 >>> import cv2

2 >>> print cv2.\_\_version\_\_

**Anaconda Installation:**

1. [Download the Anaconda installer](https://www.anaconda.com/download/#windows).
2. Optional: [Verify data integrity with MD5 or SHA-256](https://docs.anaconda.com/anaconda/install/hashes/). [More info on hashes](https://conda.io/projects/conda/en/latest/user-guide/install/download.html#cryptographic-hash-verification)
3. Double click the installer to launch.
4. Click Next.
5. Read the licensing terms and click “I Agree”.
6. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.
7. Select a destination folder to install Anaconda and click the Next button.
8. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.
9. Choose whether to register Anaconda as your default Python. Unless you plan on installing and running multiple versions of Anaconda, or multiple versions of Python, accept the default and leave this box checked.
10. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.
11. Click the Next button.
12. Optional: To install PyCharm for Anaconda, click on the link to <https://www.anaconda.com/pycharm>
13. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:
14. If you wish to read more about Anaconda Cloud and how to get started with Anaconda, check the boxes “Learn more about Anaconda Cloud” and “Learn how to get started with Anaconda”. Click the Finish button.
15. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator from the Recently added or by typing “Anaconda Navigator”. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above, then see our [Help page](https://docs.anaconda.com/anaconda/reference/help-support/).